IN THE CLAIMS:

1. (Currently amended) A method for identifying ~~non-externalized~~ hard-coded strings that require conversion, the method ~~are not hard-coded~~ comprising the computer-implemented steps of:

scanning [[a]] programming code for a first pair of string delimiters that are used to delimit text strings;

determining whether a string within said first pair of string delimiters is a path name to a resource file; and

if said string is not a path name to said resource file then flagging said string as a possible hard-coded string.

2. (Previously Presented) The method as recited in claim 1 wherein said string is not flagged as a possible hard-coded string if said string is a path name to said resource file.

3. (Currently amended) The method as recited in clam 1, wherein said programming code comprises platform-independent byte code.

4. (Original) The method as recited in clam 1, wherein said path name is a uniform resource locator.

5. (Currently amended) The method as recited in clam 1, wherein said path name is a resource ~~locator~~ bundle.

6. (Original) The method as recited in clam 1, wherein said string within said first pair of string delimiters is a path name to said resource file if said string is in a dot delimited notation.

7. (Currently amended) The method as recited in clam 1, wherein said programming code is scanned line by line until said first pair of string delimiters is identified.

8. (Currently amended) The method as recited in clam 7, wherein if there is any more programming code to be scanned after said first pair of string delimiters is identified, then the method further comprises the step of:

continuing to scan said programming code for a second pair of delimiters.

9. (Currently amended) A computer program product in a computer readable medium for identifying ~~non-externalized~~ hard-coded string that ~~are not hard-coded~~ require conversion, comprising:

programming operable for scanning [[a]] programming code for a first pair of string delimiters that are used to delimit text strings;

programming operable for determining whether a string within said first pair of string delimiters is a path name to a resource file; and

programming operable for flagging said string as possible hard-coded string if said string is not a path name to said resource file.

10. (Previously Presented) The computer program product as recited in claim 9 wherein said string is not flagged as a possible hard-coded string if said string is a path name to said resource file.

11. (Currently amended) The computer program product as recited in claim 9, wherein said programming code comprises platform-independent byte code.

12. (Original) The computer program product as recited in claim 9, wherein said path name is a uniform resource locator.

13. (Original) The computer program product as recited in claim 9, wherein said resource file is a resource bundle.

14. (Original) The computer program product as recited in claim 9, wherein said string within said first pair of string delimiters is a path name to said resource file if said string is in a dot delimited notation.

15. (Currently amended) The computer program product as recited in claim 9, wherein said programming code is scanned line by line until said first pair of string delimiters is identified.

16. (Currently amended) The computer program product as recited in claim 15, wherein if there is any more programming code to be scanned after said first pair of string delimiters is identified, then the method further comprises:

programming operable for continuing to scan said programming code for a second pair of delimiters.

17. (Currently amended) A data processing system, comprising:

a processor; and

a memory unit for storing instructions of said processor;

an input mechanism;

an output mechanism;

a bus system for coupling the processor to the memory unit, input mechanism, and output mechanism;

means for scanning [[a]] programming code for a first pair of string delimiters that are used to delimit text strings;

means for determining whether a string within said first pair of string delimiters is a path name to a resource file; and

means for flagging said string as a possible hard-coded string if said string is not a path name to said resource file .

18. (Previously Presented) The data processing system as recited in claim 17, wherein the system further comprises:

means for not flagging said string as a possible hard-coded string if said string is a path name to said resource file.

19. (Currently amended) The data processing system as recited in claim 17, wherein said programming code comprises platform-independent byte code.

20. (Original) The data processing system as recited in claim 17, wherein said path name is a uniform resource locator.

21. (Original) The data processing system as recited in claim 17, wherein said resource file is a resource bundle.

22. (Original) The data processing system as recited in claim 17, wherein said string within said first pair of string delimiters is a path name to said resource file if said string is in a dot delimited notation.

23. (Currently amended) The data processing system as recited in claim 17, wherein said programming code is scanned line by line until said first pair of string delimiters is identified.

24. (Currently amended) The data processing system as recited in claim 23, wherein if there is any more programming code to be scanned after said first pair of string delimiters is identified, then the method further comprises:

means for continuing to scan said programming code for a second pair of delimiters.